Simo Laitinen

Salesforce-kyselytyökalun toteutus HTML5-hybridisovelluksena

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

19.5.2014

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Simo Laitinen Salesforce-kyselytyökalun toteutus HTML5-hybridisovelluksena 34 sivua 19.5.2014
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Johtava konsultti Aki Teronen Lehtori Simo Silander

Tässä insinöörityössä toteutettiin Fluido Oy:lle mobiilisovellus Salesforcessa toimivalle Fluido Connect -kyselytyökalulle, jonka avulla voidaan luoda erilaisia asiakaskyselyjä ja kerätä vastauksia kyselyihin. Sovellus toteutettiin HTML5-hybridisovelluksena, joka mahdollisti sovelluksen toiminnan usealla eri käyttöjärjestelmällä ja mobiililaitteiden ominaisuuksien hyödyntämisen sovelluksessa.

Työssä käydään läpi sovelluksen tila työn alkaessa ja määritellään sovelluksen haluttu toiminta. Määrittelyn yhteydessä käydään läpi keskeiset vaatimukset, rajoitukset ja käyttötapaukset. Määrittelyn jälkeen tutkittiin sovelluksessa käytettyjä teknologioita, jotka käydään läpi omassa luvussaan.

Tämän jälkeen tarkastellaan sovelluksen arkkitehtuuria ja teknistä toteutusta, minkä yhteydessä käy ilmi, miten HTML5-sovellus, Phonegap-sovelluskehys ja Salesforce on liitetty toisiinsa ja kuinka ne kommunikoivat keskenään. Lopuksi käydään läpi sovelluksen testausta määriteltyjen käyttötapausten perusteella.

Työn lopputuloksena ei syntynyt täysin valmista sovellusta, mutta HTML5-hybridisovellusten ja tässä työssä käytettyjen teknologioiden voitiin todeta olevan erittäin käyttökelpoisia työn tilaajayrityksen tulevissa projekteissa.

Avainsanat	AngularJS, HTML5, HTML5-hybridisovellus, mobiilisovellus, Phonegap, Salesforce, REST

Author(s) Title Number of Pages Date	Simo Laitinen Implementation of Salesforce Survey Tool as an HTML5 Hybrid Mobile Application 34 pages 19 May 2014
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Aki Teronen, Principal Consultant Simo Silander, Senior Lecturer

In this thesis a mobile application was developed for Fluido LTD's Fluido Connect -survey tool which works on the Salesforce platform. The tool is used to generate customer surveys and gather replies for them. The application was developed as an HTML5-hybrid application which allows it to be easily ported on various operating systems and take advantage of mobile device features such as camera or internal storage.

The thesis begins by covering the state of the application at the beginning of the project and determines its main requirements, restrictions and use cases. The utilized technologies are also introduced.

Following this, the architecture and technical implementation of the application are examined. This demonstrates how an HTML5-application, Phonegap-framework and Salesforce integrate and communicate with each other. Finally, the testing and how the testing of the application was performed are investigated.

The thesis did not result in a fully finished application but it evidently shows that HTML5-hybrid applications, along with the other applied technologies, are very versatile and will definitely prove to be useful in future projects for the customer company.

Keywords	AngularJS, HTML5, HTML5-hybrid application, mobile application, Phonegap, Salesforce, REST

Sisällys

Lyhenteet

API — Application Programming Interface. Ohjelmistorajapinta, jonka avulla ohjelma/ohjelmisto voi keskustella muiden ohjelmien kanssa.

Asiakkuudenhallinta — Asiakaslähtöinen ajattelutapa, jossa asiakkaat ja asiakassuhteet jaotellaan niiden merkityksen mukaan.

BDD — Behaviour Driven Development. Ohjelmistokehitysmalli, jossa sovelluskehitys tapahtuu ohjelman käyttäytymisen mukaan.

CSS — Cascading Style Sheets. Merkintäkieli verkkosivujen ulkoasun määrittelemiseen.

CSV — Comma Separated Values. Yksinkertainen tekstimuoto, jossa tieto on eritelty toisistaan pilkuin ja rivinvaihdoin.

DOM-puu — Document Object Model tree. Puumainen rakenne, jonka verkkoselain luo ladatun verkkosivun HTML-lähdekoodista. Rakenne mahdollistaa sisällön dynaamisen muokkaamisen JavaScriptillä.

Istunto — Pysyvä yhteys käyttäjän ja palvelimen välillä.

JavaScript — Ohjelmointikieli jota käytetään erityisesti verkkoselaimissa dynaamisen toiminnallisuuden toteuttamiseen.

JavaScript-objekti — Yleisnimitys JavaScript-ohjelmointikielessä käytetyille muuttujille ja metodeille

JSON — JavaScript Object Notation. Yksinkertainen tekstiformaatti, jota hyödynnetään erityisesti web-sovellusten tiedonsiirrossa.

Liidi — Liidi (engl. Lead) on asiakkuudenhallintaohjelmistoissa käytetty nimitys mahdolliselle uudelle maksavalle asiakkaalle.

MVC Model-view-controller. Suunnittelumalli, jossa sovelluksen logiikka ja käyttöliittymä erotellaan toisistaan.

Salesforce-instanssi
Salesforce-pilvipalvelussa toimiva muista erillään oleva ympäristö, jota käyttää yksi tai useampi käyttäjää.

Salesforce-objekti
Salesforcen tietokannassa oleva tietokantataulukko.

SOQL Salesforce Object Query Language. Salesforcessa käytettävä kyselykieli tiedon hakemiseen tietokannasta.

XML Extensible Markup Language. Merkintäkieli, jolla kuvataan tekstin rakennetta.

1 Johdanto

Fluido Oy on 2009 vuoden loppupuolella perustettu IT-alan asiantuntijayritys. Yritys on tällä hetkellä Pohjoismaiden johtava Salesforce-konsulttiyritys, joka tarjoaa konsultoinnin lisäksi koulutuksia ja asiakkaan tarpeisiin räätälöityä ohjelmistokehitystyötä. Asiakkaisiin lukeutuu pienempien yritysten lisäksi myös isoja pörssiyhtiöitä. Asiakkaiden tarpeisiin räätälöidyn ohjelmistokehitystyön lisäksi yritys kehittää sovelluksia Salesforcen tarjoamaan AppExchange-sovelluskauppaan. Yksi sovelluskaupasta löytyvistä sovelluksista on nimeltään Fluido Connect. [1; 2.]

Fluido Connect on Salesforcen Force.com-alustalla toimiva kyselytyökalu, jonka avulla voidaan luoda erilaisia kyselyitä, vastaanottaa vastauksia kyselyihin ja analysoida saatuja vastauksia. Monipuolisten kysymystyyppien vuoksi sitä voidaan käyttää esimerkiksi markkinatutkimuksiin, asiakaspalautekyselyhin ja tapahtumien ilmoittautumisvahvistuksiin. Sovellus toimii perinteisen web-sovelluksen tapaan, eli toisin sanoen vaatii internetyhteyden toimiakseen. Tämä on havaittu ongelmaksi esimerkiksi suurissa yleisötapahtumissa, joissa matkapuhelinverkko voi olla ruuhkautunut. Toimintaa tai ulkoasua ei ole sen erityisemmin suunniteltu mobiililaitteille, vaikka käyttö tapahtuu usein matkapuhelimella tai tablet-laitteella. Sovellukselle, jonka toiminta on suunniteltu nimenomaan mobiililaitteita silmällä pitäen ja joka toimii myös offline-tilassa, oli tarvetta. [2.]

Vaihtoehtoja web-sovelluksen toteuttamiseksi mobiilisovelluksena on nykyään kolme: natiivi-, HTML5-sovellus tai hybridisovellus. Natiivisovelluksen tapauksessa kehitys tapahtuu suoraan jollekin alustalle, esimerkiksi Androidille tai iOS:lle, jolloin ohjelmistokehitys on tehtävä jokaiselle alustalle erikseen alustan tukemalla ohjelmointikielellä ja kehitystyökaluilla. HTML5-sovellus tarjoaa laitteistoriippumattoman ratkaisun, koska sovellus toimii suoraan laitteen www-selaimessa. Tällöin sovellus on suunniteltava selaimen ominaisuuksien puitteissa eikä sovellusta voi esimerkiksi asentaa laitteelle. HTML5-hybridisovellus yhdistää nämä kaksi tekniikkaa, jolloin HTML5-sovellusta voidaan suorittaa natiivisovelluksen sisällä. Tämä lähestymistapa valittiin tässä insinöörityössä läpikäytävän sovelluksen kehitykseen.

Tämä insinöörityö tehtiin Fluido Oy:lle ja työssä käydään läpi Fluido Connect -sovelluksen toteutus HTML5-hybridisovelluksena. Työn läpikäyntiin kuuluu

mobiilisovelluksen tilan kartoittaminen työn alkaessa, perehtyminen vaatimuksiin, jotka sovelluksen on täytettävä ja tarkempi selvitys sovelluksen toiminnallisesta määrittelystä. Ennen tekniseen osaan siirtymistä kerrotaan ja tutkitaan käytettyjä tekniikoita. Teknisessä osassa tutkitaan sovelluksen to arkkitehtuuria ja selitetään, miksi päädyttiin kuhunkin ratkaisuun. Testaus kuului kehitystyöhön, ja sen osalta käydään läpi ohjelmakoodin ja käyttöliittymän testaus.

2 Lähtötilanne ja toiminnallisuuden määrittely

Aloittaessani insinöörityön Fluido Connectin mobiiliversio Fluido Connect Mobile oli jo kehitteillä, mutta ei aktiivisessa kehityksessä. Ensimmäinen tehtävä oli näin ollen sovelluksen sen hetkisen tilanteen kartoittaminen. Kartoituksen yhteydessä käytiin läpi aiemmin laadittu dokumentaatio ja sovelluksen lähdekoodi. Aiempi kehitystyö oli dokumentoitu hyvin: sovelluksen toiminnalliseen ja tekniseen dokumentaatioon oli panostettu. Vaikka tekninen toteutus oli dokumentoitu hyvin ja ratkaisut olivat perusteltuja, päätettiin lähdekoodille suoritetun tarkkailun perusteella suurin osa ohjelmasta kirjoittaa uudelleen. Uuden toteutuksen myötä sovelluksen arkkitehtuuriin ja testaamiseen oli tarkoitus kiinnittää aiempaa enemmän huomiota. Tämän vuoksi sovelluksen vaatimukset, tyypilliset käyttäjät, käyttö ja käyttötilanteet, rajoitukset ja käyttötapaukset päivitettiin ajan tasalle. Fluido Connect Mobilen vaatimukset tulivat alunperin Fluidon puolelta, mutta niistä keskusteltiin ja yhteisymmärryksessä vaatimukset määriteltiin seuraavasti:

- Sovelluksen on toimittava vähintään Android- ja iOS -alustalla.
- Sovelluksen on toimittava myös ilman internetyhteyttä.
- Autentikoinnin on tapahduttava Salesforcen tarjoamia tekniikoita hyödyntäen.
- Sovelluksen on kyettävä lukemaan ja lähettämään tietoa Salesforceen.
- Sovelluksella on voitava hakea kyselyitä ja mahdollistaa kyselyihin vastaaminen.
- Käyttäjätiedot, ladatut kyselyt ja kyselyiden vastaukset on tallennettava mobiililaitteelle offline-tilan mahdollistamiseksi.
- Sovelluksen on pystyttävä käyttämään mobiililaitteen ominaisuuksia, kuten kameraa.

Sovelluksella on kahdenlaisia käyttäjiä: sovelluksen mobiililaitteellensa ladannut henkilö (esimerkiksi myyjä) ja kyselyitä täyttävä henkilö (esimerkiksi asiakas). Sovelluksen ladannut henkilö voi selata ja hakea luotuja kyselyitä, tarkastella kyselyiden vastauksia ja asettaa laitteen ottamaan vastaan vastauksia kyselyihin. Kyselyitä hallitseva käyttäjä ei pääsääntöisesti itse vastaa kyselyihin, vaan kerää vastauksia esimerkiksi tapahtumiin osallistuvilta satunnaisilta ihmisiltä. Tyypillisiä käyttötilanteita ovat esimerkiksi messut, joissa myyjä kiertelee ja kerää vastauksia messuvierailta. Mobiililaite voidaan myös asetella sille tarkoitettuun telineeseen esimerkiksi kaupan uloskäynnin oheen ja kerätä palautetta asiakkailta.

Määrittelyn yhteydessä asetettiin joitain rajoituksia sovelluksen käytölle. Rajoitukset liittyivät itse sovelluksen käyttöön tai johtuivat Salesforcen toiminnasta itsessään. Rajoitukset olivat seuraavat:

- Kyselyitä hallitsevalla käyttäjällä on oltava käyttäjätunnus Salesforce-instanssiin, jonne Fluido Connect on asennettu.
- Käyttäjällä on oltava oikeus käyttää Fluido Connectia ja siihen liittyvää tietoa.
- Mobiilisovelluksen käyttö on oltava mahdollista vain, jos käyttäjä on kirjautunut sovellukseen vähintään kerran aikaisemmin.

Sovelluksen uuden toteutuksen myötä käytiin läpi ja päivitettiin alkuperäisiä käyttötapauksia. Käyttötapauksilla kuvataan toimintoja, joita käyttäjän on voitava järjestelmällä tehdä. Niiden pohjalta voitiin luoda suoraan sovelluksen kehityksessä käytettyjä BDD-kuvauksia, jotka käydään tarkemmin läpi testausta käsittelevässä kappaleessa. Tämän työn toteutukseen valitut käyttötapaukset olivat seuraavat:

- Käyttäjä käynnistää sovelluksen ensimmäistä kertaa ja kirjautuu sovellukseen Salesforcen kautta. Mobiililaite on yhteydessä internettiin.
- Käyttäjä on käyttänyt sovellusta aikaisemmin ja kirjautuu Salesforcen kautta mobiilisovellukseen. Mobiililaitteen internet-yhteydellä ei ole väliä.
- Käyttäjä hakee kyselyitä, joihin hänellä on oikeus.
- Käyttäjä etsii kyselyitä hakusanan perusteella.
- Käyttäjä avaa kyselyn, näkee kyselyn tiedot ja vastaukset

- Käyttäjä aloittaa kyselyn täyttämisen mobiililaitteella.
- Käyttäjä tallentaa kyselyn, jonka yhteydessä kyselyn vastaukset lähetetään Salesforceen.
- Käyttäjä synkronoi synkronoimattomat kyselyjen vastaukset Salesforceen.

3 Käytetyt teknologiat

Fluido Connect Mobilen määrittelyvaiheen jälkeen tiedettiin tarkasti, miten sovelluksen tulee toimia ja mitä rajoitteita sovelluksen toimintaan liittyy. Tämän myötä voitiin siirtyä tutkimaan teknologioita, joilla sovellus on mahdollista ja järkevintä toteuttaa. Fluido Connect Mobilen teknologiavalinnoissa kiinnitettiin huomiota siihen, kuinka teknologioita oli aiemmin Fluidolla käytetty. Osa oli ollut käytössä muissa projekteissa ja myönteisten kokemusten perusteella päätettiin ottaa mukaan myös tähän työhön. Osasta taas oli vähän tai ei lainkaan kokemusta, mutta tutkimusmielessä ne otettiin mukaan ja samalla selvitettiin niiden potentiaalinen käyttö tulevaisuuden projekteissa. Mukaan on otettu HTML5-hybridisovelluksen ja tämän työn kannalta oleellisimmat teknologiat.

3.1 HTML5

HTML5:lla viitataan yleisesti kahteen asiaan: HTML-merkintäkielen viimeisimpään viidenteen versioon ja web-tekniikoilla, esimerkiksi HTML:llä, CSS:llä ja JavaScriptillä rakennettuun HTML5-sovellukseen. Merkintäkielenä sen on tarkoitus esittää ja jäsentää www-sivujen rakennetta. Aiemmista versioista eroten HTML5-standardi esittelee lukuisia uusia elementtejä, jotka mahdollistavat entistä paremmin jäsenneltyjen verkkodokumenttien määrittelemisen. Näitä elementtejä ovat esimerkiksi ”<nav>” ja ”<article>”. ”<nav>”-elementillä voidaan erotella sovelluksen navigaatiovalikko muusta sisällöstä. ”<article>”-elementillä voidaan ryhmitellä johonkin tiettyyn asiaan keskittyvä sisältö, esimerkiksi www-sivulla oleva tietokoneista kertova artikkeli, omaksi asiakokonaisuudeksi. Elementeillä pyritään ennen kaikkea tarjoamaan standardin mukainen tapa jäsennellä tietoa, mistä on apua esimerkiksi sisällön indeksointia tekeville hakuroboteille. Pelkkään sivun jäsennykseen liittyvien elementtien lisäksi HTML5 esittelee elementtejä, joiden avulla voidaan esimerkiksi toistaa

videokuvaa ja ääntä. Elementit hyödyntävät suoraan selaimen resursseja, joten erillisille lisäosille kuten Flash-laajennuksille ei ole tarvetta. [5, s. 15, 18-23.]

Uusien elementtien lisäksi HTML5 esittelee sovellusrajapintoja, joiden avulla voidaan hyödyntää selaimen ominaisuuksia. Näitä ominaisuuksia ovat esimerkiksi tiedon tallentaminen selaimen paikalliseen muistiin (engl. Web Storage) ja verkkosivun tallentaminen selaimen välimuistiin (engl. Application Cache), mikä mahdollistaa esimerkiksi verkkosivun suorittamisen ilman internetyhteyttä.

HTML5-sovellus

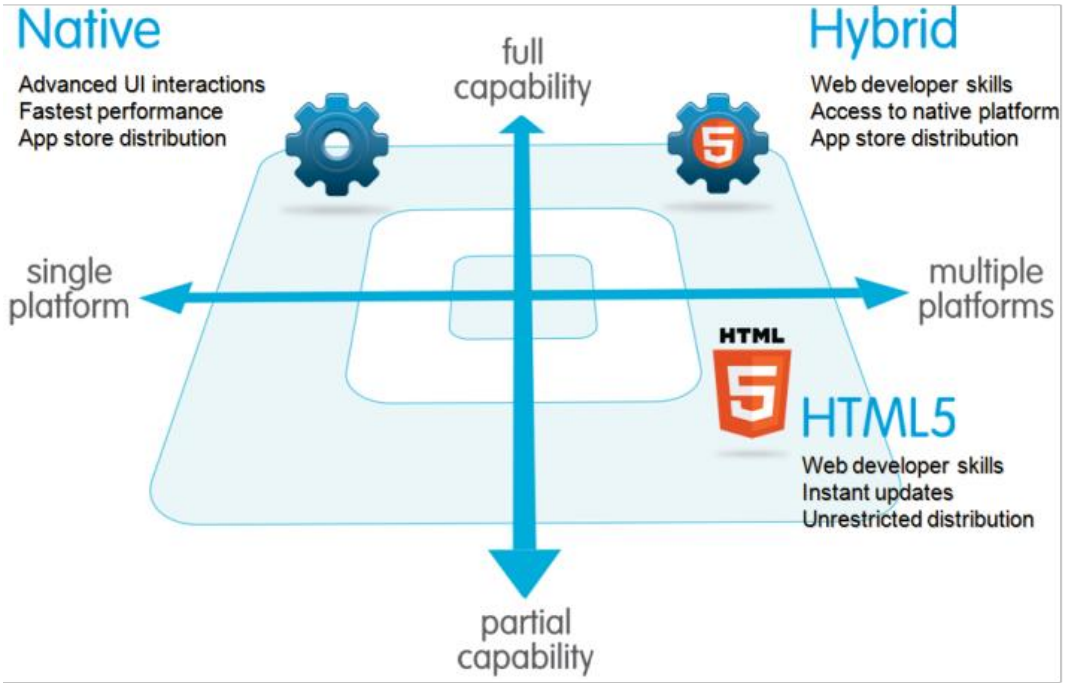
HTML5-sovelluksella tarkoitetaan selaimessa toimivaa, web-tekniikoilla rakennettua sovellusta, joka hyödyntää HTML5-ominaisuuksia. Sovellusta ajetaan käytännössä selaimen taustalta löytyvällä selainmoottorilla, joka tulkitsee ja suorittaa HTML-, CSS- ja JavaScript-koodia. Selainmoottori ei tarvitse selainta toimiakseen, vaan se voi toimia itsenäisesti tai osana jotain toista ohjelmaa, kuten esimerkiksi sähköpostisovellusta. HTML5-sovelluksen toiminta riippuu pitkälti siitä, kuinka hyvin selainmoottori tukee HTML5-standardia. Nykyisten, niin perinteisillä kuin mobiilikäyttöjärjestelmillä toimivien selaimien HTML5-tuki on melko kattava, minkä vuoksi HTML5-sovelluksia voidaan pitää alustariippumattomina, eli ne eivät ole sidoksissa tiettyyn käyttöjärjestelmään tai alustaan. Tämän ja HTML5:n tarjoaminen ominaisuuksien vuoksi HTML5-sovellukset ovat edullinen vaihtoehto natiivisovelluksille, koska usealle eri käyttöjärjestelmälle ohjelmoitavien sovelluksien sijaan sovellus voidaan ohjelmoida vain kerran. HTML5-sovellusten toiminta on kuitenkin lopulta selaimen HTML5-tuesta ja suorituskyvystä riippuvainen, joten täysin ne eivät natiivisovelluksia voi korvata. HTML5-sovelluksia ei myöskään ole mahdollista asentaa laitteille natiivisovelluksen tapaan. [3, s. 12-15; 4, s. 10.]

HTML5-hybridisovellus

HTML5-hybridisovelluksella tarkoitetaan sovellusta, jossa HTML5-sovellus toimii ja on pakattuna natiivisovelluksen sisälle. Natiivisovelluksella tarkoitetaan tässä tapauksessa millä tahansa käyttöjärjestelmällä toimivaa sovellusta. HTML5-sovelluksen ajaminen natiivisovelluksen sisällä on käytännössä mahdollista niin kutsutun web-näkymän (engl. WebView) avulla. Web-näkymä toimii verkkoselaimen tapaan, eli sen tarkoituksena on ladata tietoa verkosta tai suoraan puhelimen muistista ja suorittaa HTML5-sovelluksen

ohjelmakoodia. Käytännössä web-näkymä näyttää karsitulta verkkoselaimelta, eli siitä on poistettu osoiterivi ja muut, kuten navigointiin liittyvät painikkeet. HTML5-hybridisovelluksen mielenkiintoisin ominaisuus on kuitenkin sen tarjoama pääsy laitteen fyysisiin ominaisuuksiin, kuten esimerkiksi kameraan, sensoreihin ja suojattuun tallennustilaan. Koska HTML5-hybridisovellus on pohjimmiltaan natiivisovellus, sen asentaminen laitteelle ja jakaminen sovelluskaupoissa on mahdollista. [4, s. 12-13; 6.]

Natiivisovelluksen, HTML5-sovelluksen ja -hybridisovelluksen välisiä eroja on kuvattu kuvassa 1. Kuvassa on vertailtu sovellusten alustariippuvuutta (vaakanuoli) ja toiminnallisuuksia (pystynuoli), joita sovelluksilla on mahdollista toteuttaa.

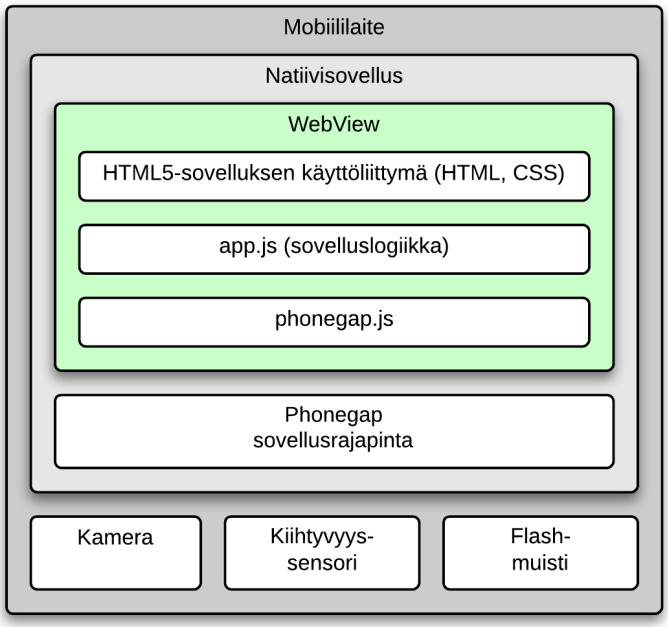


Kuva 1. Natiivisovellusten, HTML5-sovellusten ja -hybridisovellusten vertailu [7].

3.2 Phonegap

Phonegap on Adoben omistama avoimen lähdekoodin ohjelmistokehys, jonka avulla HTML5-hybridisovelluksien luominen on käytännössä mahdollista. Se perustuu Apachen Cordova-ohjelmistokehykseen, jonka avulla on niin ikään mahdollista luoda HTML5-hybridisovelluksia. Phonegap tukee useita käyttöjärjestelmiä, esimerkiksi

Androida, iOS:aa, Blackberryä ja Windows Phone 8:aa. Phonegap mahdollistaa web-sovelluksien ajamisen natiivisovelluksen tarjoaman web-näkymän kautta. Kuvassa 2 havainnollistetaan Phonegapin arkkitehtuuria kerrosarkkitehtuurikaaviona. Vihreä suorakulmio kuvaa web-näkymää, joka toimii natiivisovelluksessa (vaaleanharmaa suorakulmio). Tämän sisällä olevat kaksi ylintä suorakulmiota kuvaavat web-sovelluksen käyttöliittymää ja sovelluslogiikkaa. Phonegap.js:lla tarkoitetaan Phonegapiin sisältyvää JavaScript-kirjastoa, jonka kautta web-sovellus voi kommunikoida Phonegap-sovellusrajapinnan kanssa, joka sisältyy natiivisovellukseen. Natiivisovelluksesta voidaan edelleen kutsua mobiililaitteen (tummanharmaa laatikko) fyysisiä komponentteja, kuten kameraa, sensoreita tai laitteen sisäistä tallennustilaa. Phonegap-sovellusrajapintaa, joka itse asiassa koostuu useista lisäosista (engl. plugin), on mahdollista laajentaa valmiilla tai omilla lisäosilla.



Kuva 2. Phonegap-ohjelmistokehyksen kerrosarkkitehtuuri

Ohjelmien kääntäminen natiivisovelluksiksi on mahdollista Phonegapin tarjoamilla työkaluilla, mutta ne ovat riippuvaisia käyttöjärjestelmän kehitystyökaluista. Kääntäminen on mahdollista myös Phonegap Build -pilvipalvelussa. Palveluun voidaan ladata valmis HTML5-sovellus, jonka jälkeen palvelu huolehtii kääntämisestä halutuille käyttöjärjestelmille ja tarjoaa valmiit paketit, jotka voidaan asentaa mobiililaitteeseen. [11; 12.]

3.3 AngularJS

AngularJS on Googlen kehittämä asiakaspäässä toimiva MVC-mallin mukainen JavaScript-sovelluskehys, jolla voidaan toteuttaa dynaamisia web-sovelluksia. Web-sovelluksen logiikka voidaan toteuttaa kokonaan asiakaspäässä, jolloin perinteistä sovelluspalvelinta tarvitaan parhaimmillaan vain yksinkertaiseen tiedonvälittämiseen esimerkiksi JSON:ia käyttäen. Sovelluskehys on kehitetty erityisesti CRUD-sovelluksille, joiden toiminta siis perustuu tiedon luontiin, lukuun, päivittämiseen ja poistamiseen. Esimerkiksi lomakkeiden käsittely ja tarkistaminen (engl. form validation) kuuluu AngularJS:n perustoiminnallisuuksiin. Sovelluskehys piilottaa matalantason DOM-puun ja HTTP-pyyntöjen käsittelyn omaan abstraktiotasoon, joten sovellusta ohjelmoitaessa voidaan keskittyä itse sovelluslogiikkaan. AngularJS:n ydintoiminnallisuuksia on HTML-merkintäkielen laajentaminen niin kutsutuilla direktiiveillä. Direktiivi voi olla esimerkiksi HTML-elementti tai HTML-elementin attribuutti. AngularJS osaa yhdistää direktiivit osaksi muuta sovelluslogiikkaa, jonka vuoksi HTML:ää voidaan käyttää sovelluksen näkymien (engl. template) ohjelmointiin ilman apukirjastoja. AngularJS kytkee näkymät ja sovelluslogiikan yhtenäiseksi kokonaisuudeksi ja hoitaa tiedonvälityksen MVC-mallin mukaisten mallien ja näkymien välillä molempiin suuntiin. Kyseisestä tekniikasta käytetään nimitystä ”two way data-binding”. [13.]

AngularJS-sovelluskehyksen toiminta perustuu suurelta osin ”dependency injection”-suunnittelumallin hyödyntämiseen. Tämän suunnittelumallin ajatuksena on yksinkertaistettuna eriyttää objekti ja sen käyttämien riippuvuuksien (esimerkiksi objektin käyttämien muiden objektin) luonti ja paikantaminen toisistaan. Luokat ja komponentit eivät siis itse alusta tai hae tarvitsemiaan luokkia ja komponentteja, vaan ne saadaan parametrina alustuksen yhteydessä. Riippuvuuksien eriyttäminen logiikasta helpottaa huomattavasti testaamista, ja AngularJS tukeekin suoraan testilähtöistä ohjelmointia. [16.]

3.4 Bootstrap

Bootstrap on sovelluskehys web-sovellusten käyttöliittymien ohjelmointiin. Se on suunniteltu erityisesti käyttöliittymän skaalautuvuutta ja mobiililaitteita ajatellen. Kantavana ajatuksena on kertaalleen toteutettava käyttöliittymä, joka toimii missä vain

laitteessa ja selaimessa näytön koosta riippumatta. Käyttöliittymää ohjelmoitaessa ei näin ollen tarvitse keskittyä selainkohtaisiin rajoituksiin ja kiertoteiden etsimiseen, jotta käyttöliittymä saadaan toimimaan halutulla tavalla [15]. Sovelluskehys koostuu CSS- ja JavaScript-tiedostoista, sekä käyttöliittymäikoneista, joiden hyödyntämiseksi käyttöliittymän HTML-koodi on jäsenneltävä Bootstrapin ehtojen mukaan. Fluido Connect Mobilessa hyödynnettiin Bootstrapia, koska tarkempia käyttöliittymäsuunnitelmia ei ole tehty, eikä siihen haluttu käyttää kohtuuttomasti aikaa. Käyttöliittymän oli toimittava mobiililaitteiden näytöllä, johon Bootstrap tarjoaa valmiita, toimivia käyttöliittymäpohjia. Lisäksi se on yhteensopiva sovelluslogiikan ohjelmointiin käytetyn AngularJS:n kanssa. [14.]

3.5 Salesforce.com

Salesforce.com on Marc Benioffin vuonna 1999 perustama maailmanlaajuisesti toimiva pilvipalveluyritys, joka tarjoaa ohjelmistoja SaaS (Software as a Service) -palveluina. Yritys listautui New Yorkin pörssiin vuonna 2004, ja sen liikevaihto oli vuonna 2013 yli kolme miljardia Yhdysvaltain dollaria. Yrityksessä työskentelee noin 12 000 henkilöä ja asiakkaita on yli 100 000. Yrityksen merkittävin tuote on asiakkuudenhallintaan (engl. Customer Relationship Management) keskittyvä Salesforce [8]. Salesforcelle on tarjolla valmiita sovelluksia, joiden avulla voidaan hallita esimerkiksi yrityksen myyntiä, asiakassuhteita, markkinointia ja sen automatisointia sekä asiakaspalvelua. Näiden lisäksi sovelluksia on ladattavissa AppExchange-nimisestä Salesforce.com:in ylläpitämästä sovelluskaupasta. Ohjelmistokehitys Salesforcelle on mahdollista Force.com-alustan avulla. Alusta toimii PaaS (Platform as a Service) -periaatteen mukaisesti, eli se on käytettävissä internet-yhteyden välityksellä ja alustan toiminnan kannalta välttämättömistä laitteistoista ja ohjelmistoista vastaa Salesforce.com. [9.]

REST API

Salesforce tarjoaa REST (Representational state transfer) -rajapinnan, jonka kautta voidaan kommunikoida Force.com-alustan kanssa. REST-rajapinnalla tarkoitetaan arkkitehtuurimallia, joka kuvaa palvelimen ja asiakasohjelman välistä kommunikointia internet-yhteyden välityksellä. Kommunikointi tapahtuu lähes aina HTTP-protokollaa käyttäen, mutta käytettäväksi sopivat muutkin protokollat. REST ei nimittäin ole standardi, eikä myöskään noudata mitään standardia. Kommunikointi palvelimen ja

asiakkaan välillä tapahtuu HTTP-protokollaa käytettäessä seuraavannimisillä HTTP-pyynnöillä: GET, POST, PUT ja DELETE. GET-pyyntöä käytetään normaalisti tiedon hakuun, POST-pyyntöä tiedon lähettämiseen ja lisäämiseen, PUT-pyyntöä tiedostojen lähettämiseen ja DELETE-pyyntöä tiedon poistamiseen. Salesforcen REST-rajapinta käyttää näiden lisäksi PATCH-pyyntöä tiedon (esimerkiksi Fluido Connectissa käyttävien kyselyiden) päivittämiseen. Tärkeä osa REST-rajapinnan toteutusta on verkko-osoite (URL), jossa määritellään tieto tai toiminto, jonka palvelimen halutaan palauttavan tai suorittavan. Palautettava ja lähetettävä tieto voi olla missä muodossa vain, mutta yleisimmin käytössä on JSON- tai XML-tekstimuoto. Salesforce tukee esimerkiksi molempia muotoja. [3, s. 54; 10; 18.]

Bulk API

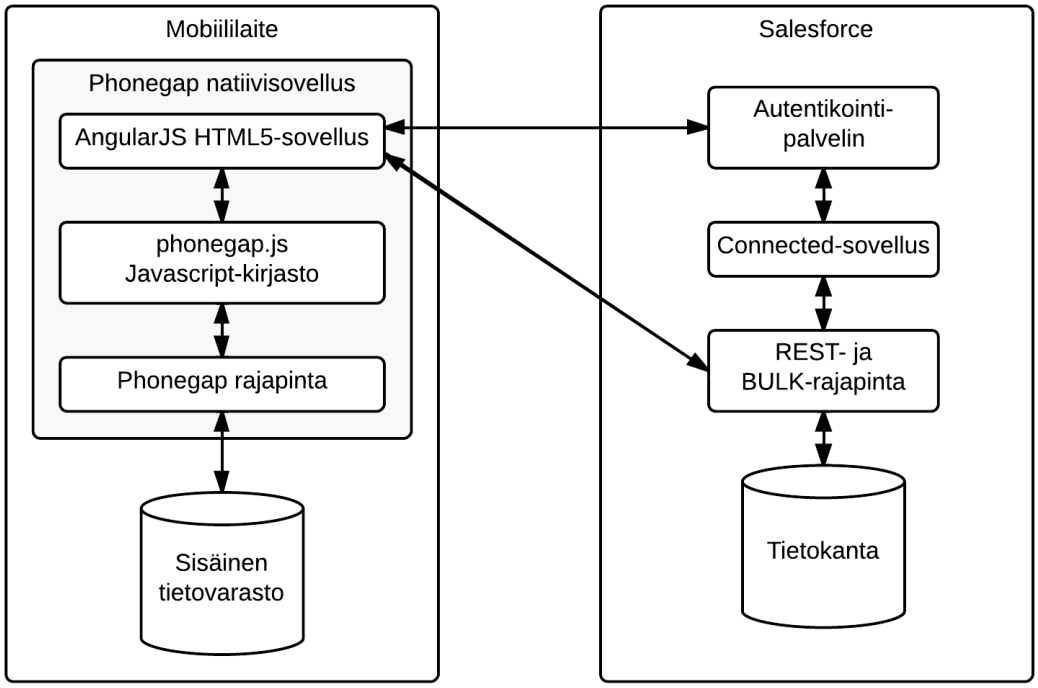
Bulk API on Salesforcen tarjoama rajapinta suurien tietomäärien hakemiseen, tallentamiseen ja poistamiseen. Sitä käytetään myös muihin rajapintoihin liittyvien rajoitusten kiertämiseksi. Rajapinta toimii edellä kuvatun REST-arkkitehtuurin mukaisesti, joskin palvelimelta palautettava ja sinne lähetettävä tieto on muotoiltava CSV:nä tai XML:änä. Salesforce käsittelee rajapinnan kautta lähetetyt tiedot asynkronisesti, eli tietoa prosessin valmistumisesta ei saada välittömästi. [17.]

4 Fluido Connect Mobilen toteutus

Fluido Connect Mobile toteutettiin HTML5-hybridisovelluksena, koska sen arveltiin olevan tehokkain keino vastata sovellukselle asetettuihin vaatimuksiin. HTML5-hybridisovelluskehykseksi valittiin Phonegap, koska sen käytöstä oli jonkin verran aiempaa kokemusta ja näin ollen sen tiedettiin soveltuvan juuri tähän projektiin. Se mahdollisti alustariippumattoman sovelluksen kehittämisen ja pääsyn mobiililaitteen fyysisiin ominaisuuksiin. Vaatimusten mukainen offline-toiminnallisuus oli mahdollinen, koska HTML5-sovellus ladattiin suoraan mobiililaitteen muistista ja käsiteltävä tieto voitiin tallentaa mobiililaitteen sisäiseen tietovarastoon. Phonegap-sovellus oli myös mahdollista asentaa puhelimeen ja sitä voitiin jakaa kunkin käyttöjärjestelmän sovelluskaupassa.

Sovelluksen ytimenä toimiva HTML5-sovellus toteutettiin AngularJS-sovelluskehyksellä. AngularJS:n valintaan vaikutti sen tuoma abstraktiotaso, joka

piilottaa esimerkiksi matalan tason DOM-puun ja HTTP-pyyntöjen käsittelyn sovelluslogiikalta. Sovelluskehystä ei ole myöskään aiemmin käytetty Fluidolla, minkä vuoksi sen käyttöä haluttiin tutkia tarkemmin. HTML5-sovellus sisältää kaiken Fluido Connect Mobileen liittyvän logiikan. Sovellus on vastuussa esimerkiksi autentikoinnista Salesforcen kanssa ja tiedon käsittelystä Force.com:in REST-rajapinnan ja mobiililaitteen tietovaraston kanssa. Tämä on nähtävissä sovelluksen arkkitehtuuria kuvaavassa kuvassa 3, joka kuvaa sovelluksen osien välistä ja sovelluksen ja Salesforcen välistä kommunikaatiota. Kuvasta nähdään esimerkiksi, että HTML5-sovelluksella on pääsy mobiililaitteen tietovarastoon ”phonegap.js” -JavaScript-kirjaston kautta, joka puolestaan kommunikoi natiivisovellukseen sisältyvän Phonegap-rajapinnan kautta. Phonegap-rajapinta puolestaan sisältää tarvittavan logiikan mobiililaitteen tietovaraston käsittelyyn. Salesforcea kuvaavan laatikon sisällä olevan ”connected”-sovelluksen merkitys käydään tarkemmin läpi autentikointia käsittelevässä luvussa.



Kuva 3. Yleiskuva Fluido Connect Mobilen arkkitehtuurista

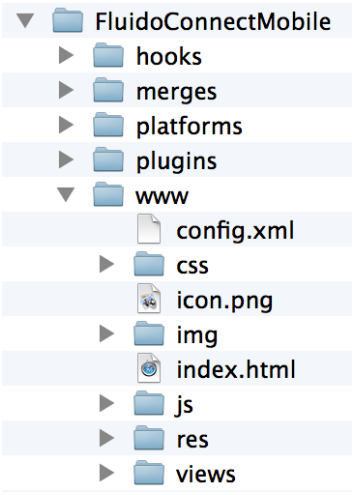
Seuraavaksi käydään läpi Fluido Connect Mobilen toteutus. Ensimmäisenä käydään läpi koko sovelluksen ja HTML5-sovelluksen rakenne, jonka jälkeen kerrotaan sovelluksen käsittelemän tiedon rakenteesta ja käsittelystä, kuten esimerkiksi offline-

toiminnallisuuden toteutuksesta. Autentikoinnin ja käyttöliittymän toteutus käydään läpi omissa alaluvuissaan. Lopuksi kerrotaan toteutetuista toiminnallisuuksista.

4.1 Sovelluksen rakenne

Fluido Connect Mobile muodostuu kokonaisuudessaan kuvassa 4 esitetyistä hakemistoista ja tiedostoista, jotka kuvaavat Phonegap-sovelluksen rakennetta. ”www”-hakemistossa on sovelluksen ydin, eli HTML5-sovellus. HTML5-sovellukseen liittyvien tiedostojen ja hakemistojen lisäksi hakemisto sisältää ”config.xml”-tiedoston ja ”res”-hakemiston, joita Phonegap käyttää kääntäessään projektin lähdekoodeja natiivisovellukseksi. ”config.xml”-tiedosto sisältää metatietoa, jossa määritellään muun muassa sovelluksen nimi, oikeudet käyttää mobiililaitteen ominaisuuksia ja ulkoisia www-palveluita. Esimerkiksi Force.com:in tarjoaman REST-rajapinnan kutsuminen ei olisi mahdollista, jos sen käyttöä ei olisi erikseen hyväksytty metatiedossa. Syynä tähän on saman alkuperän rajoitus (engl. Same-Origin policy), joka estää oletusarvoisesti web-sovelluksen kommunikoinnin tuntemattomien palveluiden kanssa [23]. ”res”-hakemisto sisältää käyttöjärjestelmäkohtaisesti natiivisovelluksen käyttämät käynnistysikonit ja aloitusruudun kuvat.

”platforms”-hakemisto sisältää natiivisovellusprojektin kullekin käyttöjärjestelmälle, jolle sovellus on käännetty. Kääntämisen yhteydessä ”www”-hakemiston sisältö käytännössä kopioidaan kunkin natiivisovellusprojektin alle. Kunkin natiivisovellusprojektin alla voidaan tehdä käyttöjärjestelmäkohtaisia muutoksia HTML5-sovellukseen, mutta joustavampi tapa on lisätä muutosta vaativat tiedostot ”merges”-hakemistoon. ”hooks”-hakemistossa voidaan määritellä komentoja, joita halutaan suorittaa esimerkiksi ennen ohjelmakoodin käännösprosessin alkamista tai sen päätteeksi. ”plugins”-hakemistoon voi puolestaan lisätä kolmannen osapuolen lisäosia, joita tosin ei tässä sovelluksessa tarvittu. [12.]



Kuva 4. Fluido Connect Mobile Phonegap-projektin tiedostorakenne

4.2 HTML5-sovelluksen rakenne

Kuten aiemmin todettiin, HTML5-sovellus toteutettiin AngularJS-sovelluskehyksellä, joka mahdollisti toteutuksen niin kutsuttuna yksisivuisena sovelluksena (engl. Single Page Application). Tällä tavalla toteutetussa sovelluksessa on vain yksi sivu (esimerkiksi index.html), joka ladataan palvelimelta (tai kuten Fluido Connect Mobilessa suoraan mobiililaitteelle asennetun sovelluksen muistista) vain kerran. Ensimmäisen sivunlatauksen yhteydessä ladataan myös sovelluksen tarvitsemat resurssit, kuten JavaScript-, tyyli- ja kuvatiedostot. Tämän jälkeen sovelluksen sisältöä päivitetään dynaamisesti JavaScript:illa. Sisältöä voidaan hakea HTTP-pyynnöin ulkoiselta palvelimelta, kuten esimerkiksi Force.com:ista REST-rajapinnan kautta. [3, s. 35-36.]

HTML5-sovelluksen rakenne jaettiin AngularJS:n tarjoamiin loogisiin osiin, joita ovat muun muassa tässäkin työssä käytetyt moduulit (engl. modules), palvelut (engl. services), kontrollerit (engl. controllers) ja näkymät (engl. templates).

Moduulit

Moduulien tarkoitus on paketoida sovelluksen muut osat, esimerkiksi palvelut ja kontrollerit, loogisiksi kokonaisuuksiksi. Moduuleista on järkevällä suunnittelulla mahdollista luoda uudelleenkäytettäviä, jolloin niitä voidaan käyttää sellaisenaan muissa projekteissa. AngularJS-sovellus voi sisältää useita moduuleja, jotka voivat

koostua toisista moduuleista. Fluido Connect Mobilen kehityksessä syntyi esimerkiksi kolme moduulia: ”fcm”, ”sf” ja ”storage”. ”fcm”-moduuli koostuu ”Route”-, ”sf”- ja ”storage”-moduuleista, ja se sisältää kaikki Fluido Connect Mobilessa käytettävät palvelut, kontrollerit ja suodattimet. ”Route”-moduuli sisältyy AngularJS-sovelluskehykseen ja sitä voidaan käyttää esimerkiksi kontrollerien ja näkymien liittämiseksi toisiinsa. ”sf”-moduulissa on toteutettu kommunikointi Force.com:in REST-rajapinnan kanssa, ja ”storage”-modulissa tiedon luku ja tallennus mobiililaitteen paikalliseen tietovarastoon. AngularJS-moduuleihin sisältyy lisäksi ”config”-metodi, jossa voidaan alustaa moduuli ennen kuin AngularJS-sovelluksen suoritus käynnistyy. Koodiesimerkki 1:ssä on kuvattu ”fcm”-modulin luonti ja sen koostuminen ”Route”-, ”sf”- ja ”storage”-moduuleista, sekä ”ui.bootstrap”-moduulista, jota käytettiin ulkoasun ohjelmointiin/taittoon (rivi 1.). Riviltä 2 alkaa moduulin alustusfunktio, jossa määritellään sovelluksen navigointilogiikka.

```
1.  var app = angular.module('fcm', ['ngRoute', 'sf', 'storage',
                                                    'ui.bootstrap']);
2.  app.config(['$routeProvider', function($routeProvider){
3.      $routeProvider.
4.      when('/', {
5.          templateUrl: 'views/home.html',
6.          controller: 'homeCtrl'
7.      }).
8.      when('/surveys', {
9.          templateUrl: 'views/surveys.html',
10.         controller: 'surveysCtrl'
11.     }).
12.     // config-metodi jatkuu
```

Koodiesimerkki 1. Fluido Connect Mobilen päämoduulin esittely JavaScriptissä.

Ennen kuin AngularJS aloittaa sovelluksen alustamisprosessin, jossa yhdistetään esimerkiksi kontrollerit näkymiin, on sille kerrottava, mitä moduulia sen tulisi käyttää sovelluksen päämoduulina. Tämä voidaan tehdä automaattisesti asettamalla johonkin ”index.html”-tiedostossa olevaan elementtiin ”ng-app”-direktiivi, johon puolestaan määritellään moduulin nimi. AngularJS tulkitsee ainoastaan elementin sisällä olevat elementit osaksi sovellusta, joten direktiivi kannattaa asettaa elementille, joka on mahdollisimman juuressa. AngularJS aloittaa sovelluksen alustamisen ensimmäisen sivunlatauksen yhteydessä heti ”ng-app”-direktiivin löydettyään. On kuitenkin tapauksia, joissa sovellusta ei haluta alustaa heti sivunlatauksen yhteydessä. Esimerkiksi Fluido Connect Mobilen on odotettava ennen alustamista, että Phonegap-kirjasto on latautunut ja alustettu. Phonegap lähettää onnistuneen alustuksen

yhteydessä ”deviceready”-viestin, joka voidaan havaita ja aloittaa AngularJS-sovelluksen alustaminen (koodiesimerkki 2).

```
1.  document.addEventListener('deviceready', function onDeviceReady() {
2.    angular.bootstrap(document, ['fcm']);
3.  }, false);
```

Koodiesimerkki 2. Sovelluksen päämoduulin alustus Phonegapin alustuksen ollessa valmis.

Palvelut

Palvelut ovat johonkin tiettyyn asiaan erikoistuneita komponentteja, jotka määritellään moduulin sisällä ja jotka ovat moduuliin sisältyvien osien käytettävissä. Moduulien tapaan palvelut voivat koostua muista palveluista. Palveluissa määritellään tyypillisesti osa kehitettävän sovelluksen business-logiikasta. Kehitetyssä sovelluksessa ”fcm”-moduulin avainlogiikka on sijoitettu palveluihin, jotka ovat vastuussa muun muassa autentikointiin ja kyselyjen sekä virheviestien hallintaan liittyvästä logiikasta. Oheisessa koodiesimerkissä 3 on kuvattu ”survey”-palvelun toiminnallisuutta. Palvelu luodaan koodiesimerkissä 1 esitetyn ”app”-objektin sisään, joka on siis sovelluksen päämoduuli, eli ”fcm”-moduuli.

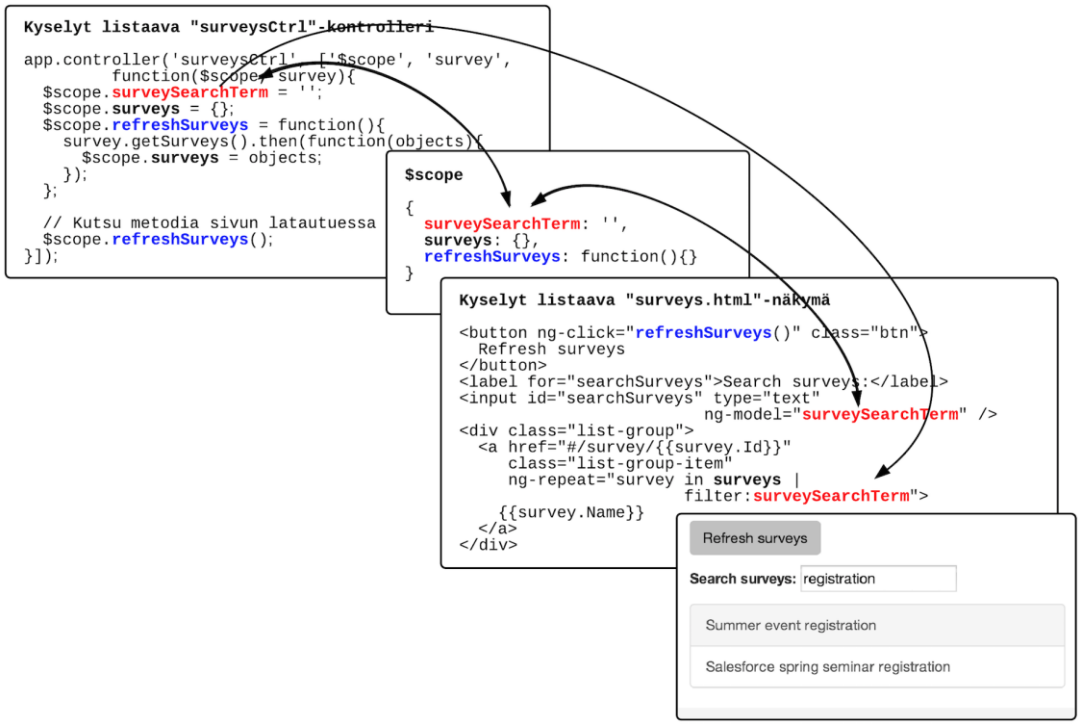
```
1.  app.service('survey', ['$q', 'auth', 'sf.rest',
2.           function($q, auth, sfApi){
3.       this.currentSurvey = {};
4.       this.surveys = {};
5.       this.surveyResponses = {};
6.
7.       this.getSurvey = function(surveyId){
8.           var def = $q.defer();
9.           sfApi.getObjectById('fluidoconnect1__Survey__c',
10.                  surveyId).then(function(data){
11.              this.currentSurvey = data;
12.              def.resolve(data);
13.          }, function(data){
14.              def.reject(‘error’);
15           });
16.          return def.promise;
17.      };
18.      // Palvelun toteutus jatkuu..
```

Koodiesimerkki 3. Yksinkertaistettu näkymä “survey”-palvelun toteutuksesta

Kontrollerit ja näkymät

Kontrollereita käytetään AngularJS:ssä näkymiltä tulevien käskyjen käsittelyyn. Ne voivat koostua useista palveluista, joihin suurin osa kontrolleriin liittyvästä business-logiikasta yleensä ulkoistetaan. Näkymät puolestaan vastaavat tiedon näyttämisestä ja vastaanottamisesta käyttäjältä. Kontrollerit ja näkymät voidaan liittää yhteen joko lähdekoodissa tai näkymässä ”ng-controller”-direktiivillä. Koodiesimerkissä 1 riviltä 8 lähtien on kuvattu ”surveyCtrl”-kontrollerin ja ”surveys.html”-näkymän liittämistä yhteen ”surveys”-sivulle navigoitaessa.

Näkymät, kontrollerit ja kontrollereissa määriteltävät mallit, jotka ovat JavaScript-objekteja, toteuttavat yhdessä MVC-arkkitehtuuria. Kuvassa 5 on esitetty tiedonvälitystä HTML5-sovelluksen kyselyitä listaavalla sivulla. Käyttäjän tekemät muutokset, esimerkiksi hakusanan syöttäminen hakukenttään, päivittyvät reaaliaikaisesti kontrollerille. Samoin kontrollerissa tehtävät muutokset näkyvät välittömästi käyttäjälle. AngularJS käyttää tästä kahdensuuntaisesti tiedon päivittämisestä aiemmin mainittua nimitystä ”two-way data binding”. Kuvan esittämässä tilanteessa hakukenttään tehtävät muutokset vaikuttavat suoraan punaisella merkittyyn ”surveySearchTerm”-malliin. Kyseistä mallia käytetään myös suodattimena (engl. filter) kyselyiden listauksessa, joten käyttäjä näkee haun tulokset välittömästi. Kontrollerin ja näkymän väliin merkitty ”$scope”-taso on AngularJS:n ylläpitämä, DOM-puun rakennettava vastaava looginen tietorakenne joka havaitsee näkymässä ja kontrollerissa tapahtuvat muutokset. [19.]



Kuva 5. Kyselyiden listaukseen liittyvä yksinkertaistettu kontrolleri ja näkymä. [19.]

Tiedon välitys sovelluksen eri osien välillä

Tiedon välitykseen kontrollereiden, palveluiden ja moduulien välillä hyödynnettiin niin kutsuttuja ”deferred”- ja ”promise”-objekteja, joiden tarkoitus on helpottaa asynkronisuuden hallintaa. Niiden avulla asynkroniset metodit voitiin ohjelmoida näennäisesti samaan tapaan kuin synkroniset metodit, mikä helpottaa ohjelmakoodin luettavuutta ja ylläpidettävyyttä. Asynkronisella metodilla tarkoitetaan yleisesti metodia, joka jää suorittamaan taustalle jotakin tiettyä toimintoa (esimerkiksi HTTP-pyyntöä) metodikutsun päätyttyä ja palauttaa ehkä joskus jotain tietoa toiminnon onnistumiseen liittyen. Synkronisen metodin valmistumista puolestaan jäädään odottamaan ja metodin lopussa tiedetään saman tien, onnistuiko haluttu toiminto. Tieto asynkronisen metodin valmistumisesta saadaan JavaScriptissä normaalisti metodille parametrina annettavalla ”callback”-metodilla, jota kutsutaan toiminnon valmistuessa. ”deferred”-objekti poistaa tarpeen edellä mainituille ”callback”-metodeille. Se luodaan asynkronisen metodin sisällä (kuten koodiesimerkissä 3 rivillä 8.) ja sitä voidaan pitää eräänlaisena työnjohtajana, joka osaa kertoa, onnistuiko metodin ajama toiminto. ”deferred”-objektin tila, eli tieto metodin onnistumisesta saadaan palautettua metodin päätteeksi ”promise”-objektin avulla. ”promise”-objektilla on kolme tilaa, joista kaksi ilmaisevat toiminnon

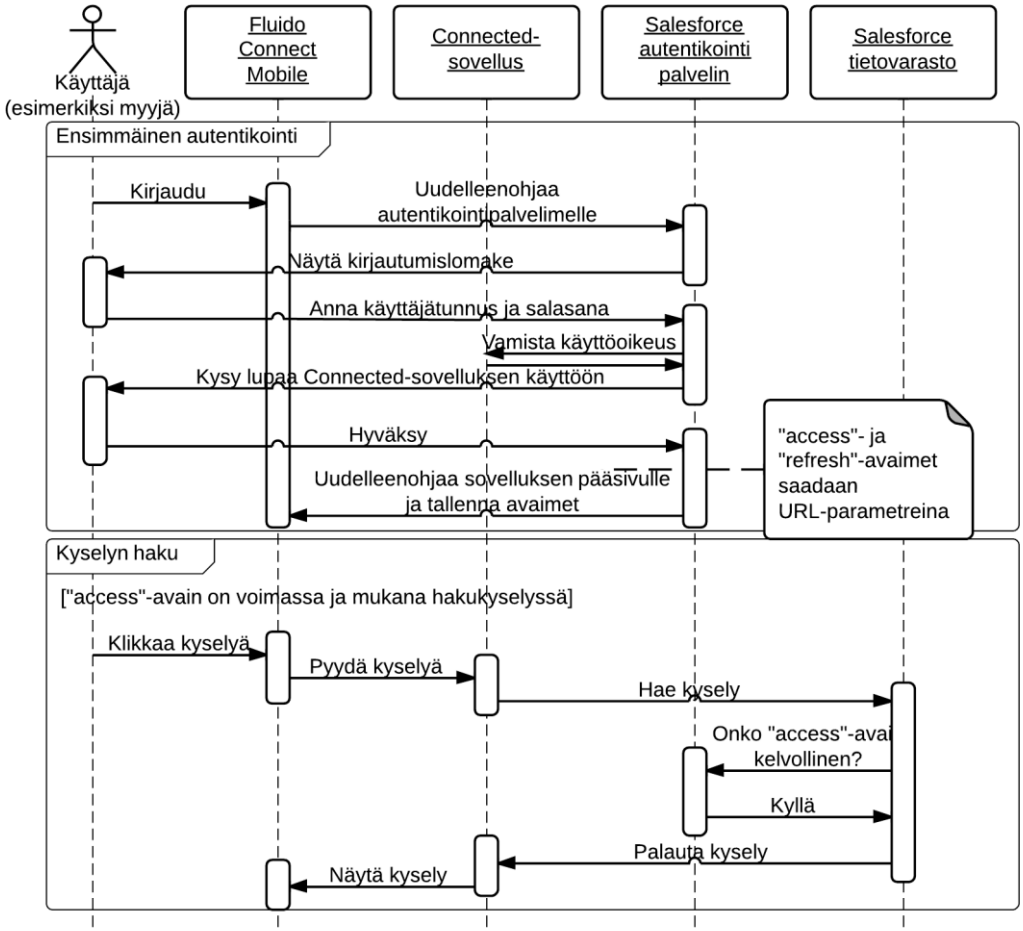
onnistumista ja epäonnistumista. Kolmas tila kertoo ”promise”-objektin vielä odottavan tietoa toiminnon onnistumisesta. [22.]

4.3 Autentikointi

Autentikoinnilla tarkoitetaan yleisesti identiteetin varmistamista. Identiteetillä tarkoitetaan esimerkiksi käyttäjää tai palvelua, jolla on jokin tunnistettavissa oleva ominaisuus. Esimerkiksi Salesforcessa käyttäjän tunnistamiseen käytetään käyttäjätunnusta, salasanaa ja joissakin tapauksissa Salesforcen automaattisesti luomaa salaista avainta. Identiteetin varmistamisen perusteella identiteetille voidaan sallia pääsy johonkin tiettyyn asiaan tai tietoon, esimerkiksi Salesforce-objektiin tai -sovellukseen. [20.]

Fluido Connect Mobile käyttää autentikointiin Salesforcen tukemaa OAuth2-protokollaa. OAuth2-protokolla mahdollistaa autentikoinnin asiakassovellukseen ilman erillistä käyttäjätunnusta tai salasanaa. Esimerkiksi Fluido Connect Mobilessa käyttäjä käyttää tunnistautumiseen Salesforce-käyttäjätunnuksiaan. Autentikoinnista vastaa erillinen autentikointipalvelin, joka palauttaa onnistuneen autentikoinnin yhteydessä avaimia, joilla taataan pääsy tietovarastoon. Avaimia on kaksi: ”access token” ja ”refresh token”. ”access token”-avainta voidaan käyttää suoraan palvelimelle tehtävissä HTTP-pyynnöissä, ja sen avulla palvelin tietää, mihin tietoon käyttäjällä on oikeus. Se on lyhytikäinen, istunnon ajan voimassa oleva avain, joka vanhenee yleensä muutamien minuuttien tai tuntien kuluessa. ”refresh token”-avaimella voidaan uusia ”access token”-avain. Avain on pitkäikäinen, eikä välttämättä vanhene koskaan, joten sen avulla voidaan taata pääsy sovelluksen tarvitsemiin resursseihin ensimmäisen onnistuneen autentikoinnin jälkeen. Jotta autentikointipalvelin osaisi määritellä, mihin resursseihin käyttäjällä on oikeus, on autentikointipalvelimen jollain tapaa tunnistettava asiakassovellus. Salesforcessa tunnistautumiseen vaaditaan niin kutsuttu ”Connected”-sovellus, jossa määritellään mitä tietoa sovelluksen on oikeus käsitellä. ”Connected”-sovelluksella on aina OAuth2-protokollan mukainen yksilöllinen ”Consumer”-avain, jota käytetään autentikoinnin yhteydessä. Kyseisessä sovelluksessa on määriteltävä myös osoite, jonne käyttäjä ohjataan onnistuneen autentikoinnin yhteydessä (engl. callback URL). [3, s. 371-372; 21.]

Kuvassa 6 on esitetty OAauth2-protokollan mukainen autentikointiprosessi Fluido Connect Mobilen näkökulmasta, sekä esimerkkikysely, jossa haetaan tietoa Force.com:ista onnistuneen autentikoinnin yhteydessä saatuja avaimia käyttäen. Kuvasta on jätetty selkeyden vuoksi pois virhetilanteiden käsittely. Kuvassa esitetyssä ensimmäisessä autentikoinnissa kuvataan, miltä kirjautumisprosessi näyttää käyttäjän näkökulmasta. Kuvasta nähdään, että käyttäjä ohjataan pois itse sovelluksesta Salesforcen kirjautumissivulle, jolla käyttäjän on kirjauduttava ja onnistuneen kirjautumisen yhteydessä annettava edellä käsitellylle ”Connected”-sovellukselle lupa käyttää käyttäjän tietoja. Kirjautumisprosessin onnistuttua käyttäjä ohjataan takaisin sovelluksen pääsivulle.



Kuva 6. Sekvenssikaavio autentikointiprosessista Fluido Connect Mobilen näkökulmasta.

Kuvassa 6 esitettyä uudelleenohjausta Fluido Connect Mobilen kirjautumissivulta Salesforcen autentikointipalvelimelle ei voitu toteuttaa sellaisenaan. Syynä tähän on

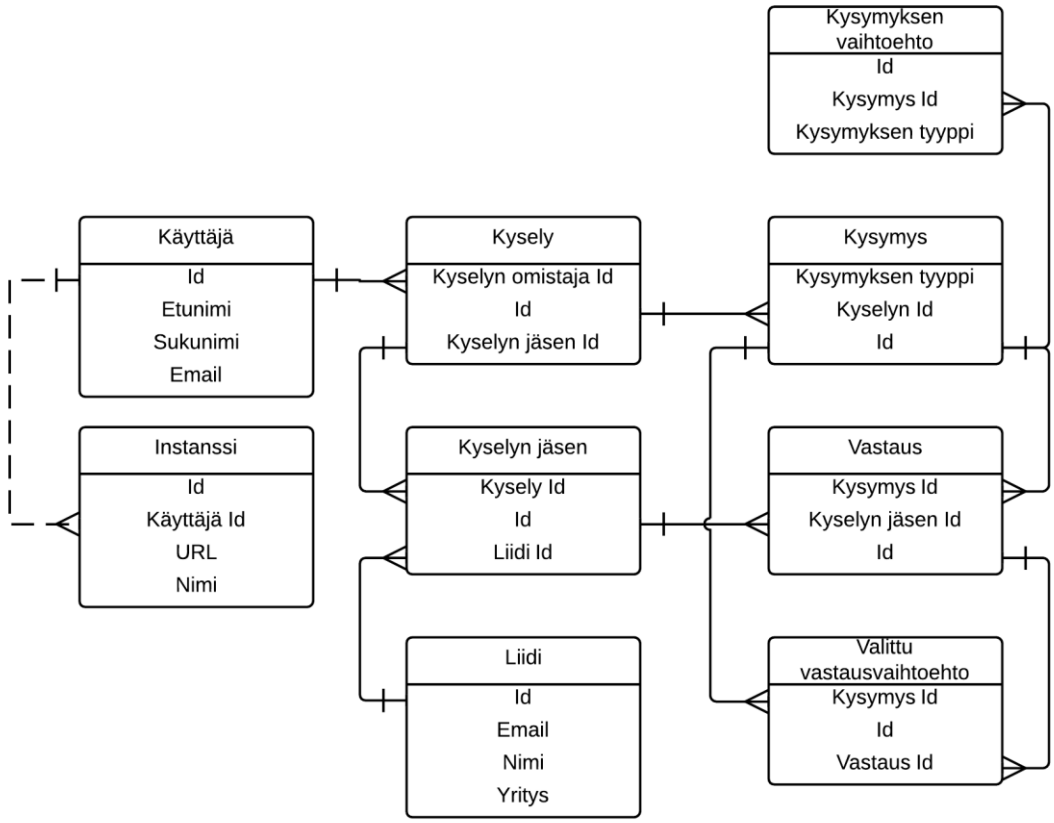
sovelluksen käyttämä web-ikkuna, joka käyttää HTTP-protokollan sijaan ”file”-protokollaa. ”file”-protokolla on tässä tilanteessa käytössä web-ikkunan sisällön lataamiseen laitteen paikallisesta muistista ulkoisen palvelimen sijaan. Salesforcen ”Connected”-sovellus ei kuitenkaan hyväksy ”file”-protokollan käyttöä osoitteessa, johon käyttäjä ohjataan onnistuneen kirjautumisprosessin päätteeksi. Käyttäjää ei siis olisi voitu ohjata takaisin Fluido Connect Mobile sovellukseen. Ongelma ratkaistiin Phonegap-sovellusrajapintaan sisältyvällä ”InAppBrowser”-lisäosalla. Lisäosa mahdollistaa ponnahdusikkunan luomisen web-ikkunan sisällä ja rajapinnan, jonka kautta ponnahdusikkuna voidaan sulkea ja tarkastella ponnahdusikkunan URL:ää. Todellisen uudelleenohjauksen sijaan Salesforcen autentikointisivu voidaan siis avata ponnahdusikkunassa ja tarkastella URL:n perusteella, koska käyttäjä on onnistunut autentikoinnissa. Autentikoinnin voidaan katsoa onnistuneen, kun URL:stä on löydettävissä ”access_token”-parametri, jonka Salesforcen autentikointipalvelin lisää URL:n perään onnistuneen autentikoinnin yhteydessä. [21.]

OAuth2-protokollan mukaisen autentikoinnin lisäksi Fluido Connect Mobile käyttää niin sanottua PIN-koodi-autentikointia, jossa kirjautumiseen vaaditaan käyttäjätunnuksen lisäksi käyttäjän määrittelemä PIN-koodi. Kyseinen autentikointitapa mahdollistaa sovellukseen kirjautumisen myös silloin, kun puhelin ei ole yhteydessä verkkoon. PIN-koodin käyttö ei ole pakollista, mutta se on ainoa tapa kirjautua sovellukseen tilanteissa, joissa mobiililaitteelta löytyy useampi käyttäjä ja laite on offline-tilassa. Käyttäjä voi asettaa PIN-koodin ensimmäisen, OAuth2-protokollan mukaisen kirjautumisprosessin jälkeen.

4.4 Sovelluksen käsittelemä tieto

Fluido Connect Mobile käsittelee pääosin samaa tietoa kuin Salesforceen asennetteva Fluido Connect. Salesforce toimii sovelluksen näkökulmasta ulkoisena tietokantana, josta tietoa haetaan ja jonne se lopulta tallennetaan. Sovellus käyttää lisäksi omaa mobiililaitteen paikallista tietovarastoa tiedon tallentamiseen, mikä mahdollistaa tiedon käyttämisen offline-tilassa. Paikallista eli sovelluslogiikassa käsiteltävää tietoa käsitellään yksinkertaisina JavaScript-objekteina. Objektien kenttien nimet vastaavat Salesforcessa määriteltyjen objektien kenttien nimiä, mikä esimerkiksi helpottaa kommunikointia REST-rajapinnan kanssa, koska tietoa ei tarvitse missään välissä muuntaa toiseen muotoon. Kuvassa 7 on esitetty Fluido Connect Mobilen käsittelemä

tieto ER-kaaviona, josta nähdään käsiteltävän tiedon eli objektien väliset riippuvuudet. Kaavioon on selkeyden vuoksi merkitty näkyviin vain pakolliset ja objektien välisiä relaatioita kuvaavat kentät.



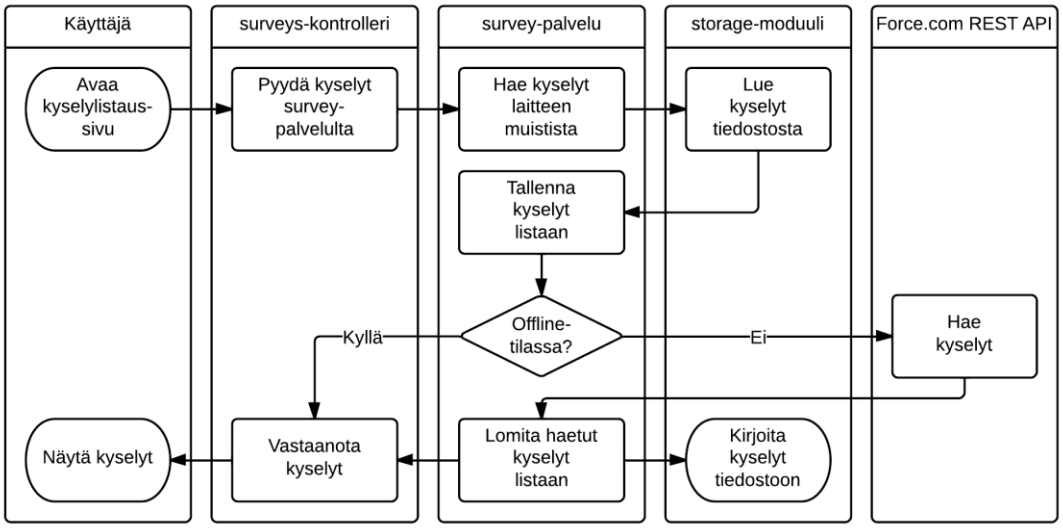
Kuva 7. Fluido Connect Mobilen tietokantamalli

Kuvassa esitetyt objektit Instanssi-objektia lukuun ottamatta, jota käytetään vain sovelluksen paikallisessa tietokannassa, kuvaavat objektien välisiä suhteita niin Salesforcessa kuin sovelluksen paikallisessa tietokannassa. Instanssi-objekti pitää kirjaa Salesforce-instansseista, joihin käyttäjän on mahdollista kirjautua. Käyttäjällä voi olla useita eri instansseja sovellukseen tallennettuna. Kirjautumisen yhteydessä valittu instanssi määrää, että vain siihen liittyvät kyselyt ovat istunnon aikana käytössä. Käyttäjä-objekti vastaa Salesforce-käyttäjää, jollainen Fluido Connect Mobileen kirjautuvalla käyttäjällä on oltava. Kyselyt koostuvat yhdestä tai useammasta kysymyksestä, joilla voi puolestaan kysymystyypistä riippuen olla kysymykseen liittyviä vaihtoehtoja. Kullakin käyttäjällä voi olla useita kyselyitä, jotka ovat näkyvillä vain käyttäjälle itselleen. Kuten kaaviosta käy ilmi, kyselyihin voivavat vastata vain kyselyn

jäsenet. Käytännön tilanteissa kyselyihin vastaavaa henkilöä ei ole ennalta liitetty kyselyn jäseneksi, vaan vastausta lähetettäessä hänestä luodaan liidi eli mahdollinen maksava asiakas. Kyselyn jäsenten vastaukset tallennetaan erilliseen vastaus-objektiin. Kuhunkin vastaus-objektiin voi liittyä vastauksen vaihtoehtoja, mikäli kyseessä on kysymystyyppi, joka vaatii erillisiä vaihtoehtoja.

4.4.1 Tiedon paikallinen käsittely

Fluido Connect Mobilen oli voitava tallentaa tietoa paikallisesti vaaditun offline-toiminnallisuuden vuoksi. Tiedon tallentamiseen päätettiin käyttää mallia, joka tallentaa oletusarvoisesti kaiken sovelluksessa luodun ja REST-rajapinnan kautta haetun tiedon laitteen muistiin. Tätä lähestymistapaa on esitetty prosessikaavion muodossa kuvassa 8, jossa kuvataan kyselyiden hakua kyselyitä listaavalle sivulle tultaessa. Kuvasta voidaan lukea, että kyselyt haetaan aina laitteen muistista, minkä jälkeen (mikäli laite on yhteydessä verkkoon) ne lomitetaan REST-rajapinnasta haettujen kyselyiden kanssa. Lomittamisella tarkoitetaan tässä yhteydessä kyselyiden vertailua keskenään, jonka perusteella ”survey”-palvelun listaan ja puhelimen muistiin tallennetaan vain uniikit kyselyt. Konfliktitilanteessa, jossa REST-rajapinnan kautta haetun ja puhelimen muistiin tallennetun kyselyn tunniste (engl. id) on sama, mutta sisältö eri, käytetään aina REST-rajapinnan kautta haettua kyselyä. Ratkaisu ei ole ongelmaton ja voi tuottaa epäjohdonmukaisuuksia esimerkiksi tilanteissa, joissa kyselyyn löytyy laitteelta vastauksia, joita ei ole vielä viety Salesforceen.



Kuva 8. Kyselyiden hakua kuvaava prosessikaavio.

Tiedon paikallinen tallentaminen toteutettiin Phonegapin tarjoaman ”File”-rajapinnan avulla. Rajapinta mahdollistaa tiedostojen lukemisen ja kirjoittamisen mobiililaitteen suojattuun tietovarastoon (engl. internal storage). Kullekin käyttäjälle luodaan ensimmäisen autentikoinnin yhteydessä oma hakemisto, joka sisältää sovelluksen käsittelemien objektien mukaiset tiedostot. Esimerkiksi ”survey”-tiedosto sisältää kaikki käyttäjän näkemät kyselyt. Tieto tallennetaan tiedostoihin JSON-muodossa, joka muodostetaan esimerkiksi kyselyiden tapauksessa ”survey”-palvelun ”surveys”-objektista. Haittapuolena tällaisessa tiedostopohjaisessa tiedon käsittelyssä on esimerkiksi tiedostonluvun ja -kirjoituksen hitaus tietomäärän kasvaessa.

4.4.2 Tiedonvälitys Salesforcen ja mobiilaitteen välillä

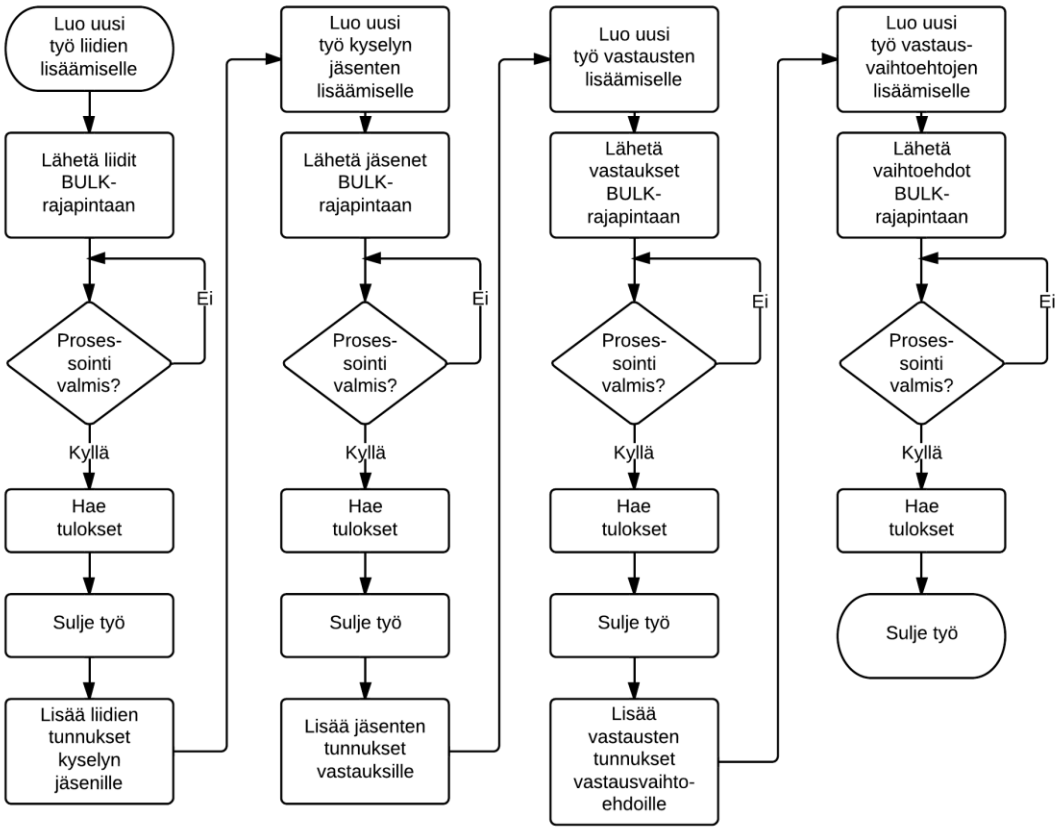
Tiedonvälitykseen mobiililaitteen ja Salesforcen välillä käytettiin Force.com-alustan tukemaa REST- ja BULK-rajapintaa. REST-rajapintaa käytettiin pääosin tiedon hakemiseen ja BULK-rajapintaa tiedon päivittämiseen ja luomiseen. Syynä tähän kahden rajapinnan käyttämiseen oli REST-rajapinnan rajoitteet tiedon luonnissa. Rajapinnan kautta on nimittäin mahdollista päivittää/luoda vain yksi Salesforce-objekti yhden HTTP-pyynnön aikana. Jokainen rajapinnan kautta tehtävä kysely kuluttaa Salesforce-instanssin päivittäin käytössä olevia rajapintakutsuja. Jos rajapintakutsujen määrä saavuttaa päivittäisen maksimimäärän, lakkaa rajapinta käsittelemästä uusia pyyntöjä.

REST-rajapinnan kutsuminen toteutettiin käytännössä ”sf”-moduulin sisältyvässä ”sf.rest”-palvelussa, joka käyttää kommunikointiin AngularJS:ään sisältyvää ”$http”-palvelua. ”sf.rest”-palvelu sisältää useita metodeja, jotka mahdollistavat tiedon kyselyn tiettyjen kenttien tai objektin nimen tai tunnuksen perusteella. Koodiesimerkissä 4 on kuvattu kaikki objektit palauttavan metodin REST-rajapintaa kutsuvaa osaa. Kyselyssä on käytetty Salesforcen SOQL-kyselykieltä, joka mahdollistaa objektien ja objektien tiettyjen kenttien palauttamisen suoraan tietokannasta.

```
1.  $http({
2.     method: 'GET',
3.     url: sfClient.instanceUrl+'/services/data/'+sfClient.apiVersion+
4.                               '/query/?q=SELECT+Id,Name+FROM+'+objName,
5.     headers: { 'Authorization':'Bearer '+sfClient.accessToken }
6.  })
```

Koodiesimerkki 4. Yksinkertaistettu metodi

Tiedon hakuoperaatioiden ollessa melko yksinkertaisia, jouduttiin tiedon lisäämisessä ja päivittämissä näkemään hieman enemmän vaivaa. BULK-rajapinnalle ei voi suoraan lähettää tietoa käsiteltäväksi, vaan ennen tätä Salesforceen on luotava työ (engl. job), joka on vastuussa tiedon prosessoinnista. Työ voi koskea tiedon lisäämistä, päivittämistä, poistoa tai lataamista, ja se määritellään työn luonnin yhteydessä. Tämän jälkeen rajapintaa voidaan kutsua työn tunnuksella ja CSV-muotoon muotoiluilla objekteilla, jotka jaetaan erikseen prosessoitaviin eriin (engl. batches). Prosessointi tapahtuu asynkronisesti, joten ainoa tapa seurata työn etenemistä on kutsua työtä luonnin yhteydessä saadulla tunnuksella. Kun prosessointi on päättynyt, voidaan prosessin tulokset hakea erikseen työn tunnuksen perusteella. Tuloksista nähdään objektikohtaisesti, onnistuiko haluttu operaatio ja samalla saadaan esimerkiksi objektien yksilölliset tunnukset, joita voidaan käyttää relaatioiden hallintaan. Työn valmistuttua se kannattaa sulkea, koska Salesforce rajoittaa yhtäaikaisten töiden määrää. Fluido Connect Mobilessa anonyymisti kyselyihin vastanneiden vastausten vieminen Salesforceen BULK-rajapinnan kautta toimii kokonaisuudessaan kuvassa 9 esitetyn logiikan mukaisesti. Kuvassa käsiteltävien objektien (liidien, kyselyn jäsenten, vastausten ja vastausvaihtoehtojen) oletetaan löytyvän mobiililaitteelta ennen prosessin alkua. Töiden suoritusjärjestyksellä on väliä, koska objektien väliset relaatiot on määriteltävä yksilöllisten Salesforcen määrittelemien tunnisteiden perusteella. Kyselyn jäseniä ei voida esimerkiksi luoda ennen liidien luontia, koska liidi-relaatio on tietomallin mukaan pakollinen kenttä.



Kuva 9. BULK-rajapinnan kautta tapahtuva kyselyiden vastausten luominen Salesforceen.

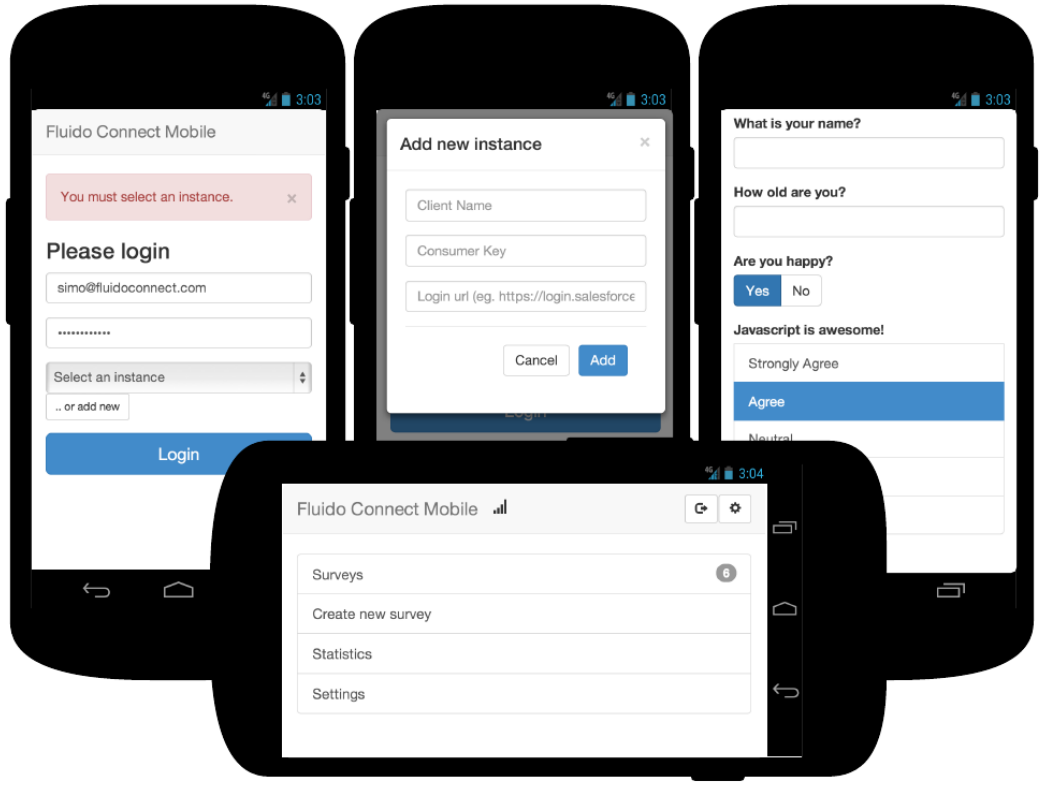
Sovelluksen käyttäjän kannalta BULK-rajapinnan käyttö ei ole paras vaihtoehto, koska BULK-työn käynnistymiseen saattaa mennä useita minuutteja. Käyttöliittymässä on siis selkeästi kerrottava kunkin työn eteneminen käyttäjälle.

4.5 Käyttöliittymän toteutus

Skaalautuva ja erityisesti mobiililaitteiden näytöllä toimiva käyttöliittymä oli yksi sovelluksen vaatimuksista. Tällaisen käyttöliittymän toteutukseen liittyy haasteita, joita aiheuttavat esimerkiksi selainten vaihtelevat tavat tulkita CSS:ää. Esimerkiksi pienet muutokset jonkin selaimen muodostamaan ulkoasuun voivat aiheuttaa arvaamatonta toimintaa toisilla selaimilla. Käyttöliittymän toteutukseen löytyy kuitenkin apukirjastoja ja jopa sovelluskehyksiä. Toteutetussa sovelluksessa käyttöliittymän rakentamiseen käytettiin Bootstrap-sovelluskehystä ja AngularUI-projektin tarjoamaa ”ui.bootstrap”-moduulia Bootstrapin ja AngularJS:n yhdistämiseen. ”ui.bootstrap”-moduuli tarjoaa

komponentteja suoraan AngularJS:n ymmärtäminä direktiiveinä. Esimerkiksi ”<alert>”-direktiivillä voidaan näyttää ohjelman toimintaan liittyviä ilmoituksia, kuten esimerkiksi ilmoitus onnistuneesta kyselyiden vastausten luontiprosessista Salesforceen.

Käyttöliittymä saatiin toteutettua vaatimusten mukaiseksi kokonaisuudessaan Bootstrapilla ja ”ui.bootstrap”-moduulilla, eli omia tyylimäärityksiä ei tarvinnut kirjoittaa riviäkään. Tämä oli mahdollista osittain käyttöliittymän yksinkertaisuuden vuoksi, jota toisaalta Fluidon puolelta toivottiin. Käyttöliittymä koostuu pääosin yllä mainituista ilmoituksista, modaalisista ikkunoista, nappuloista, listoista ja tekstikentistä, joita on nähtävissä kuvassa 10. Navigointiin sivujen välillä käytettiin yksinkertaisia painikkeita.



Kuva 10. Otteita Fluido Connect Mobilen käyttöliittymästä.

4.6 Toteutettu toiminnallisuus

Sovelluskehityksen aikana pyrittiin toteuttamaan mahdollisimman monta käyttötapauksissa määriteltyä toiminnallisuutta. Pääpaino oli autentikoinnilla,

kyselyiden listaamisella ja hakemisella, sekä kyselyihin vastaamisella ja tiedon välittämisellä Salesforceen.

Autentikointi

Kuten autentikointia käsittelevässä kappaleessa todettiin, Fluido Connect Mobile hyödyntää Salesforcen OAuth2-protokollaa käyttäjän tunnistamiseen. Kyseisen autentikointiprosessin lisäksi toteutettiin PIN-koodi-autentikointi, joka mahdollisti kirjautumisen myös laitteen ollessa offline-tilassa.

Kyselyiden listaus ja haku

Kyselyiden listaus on toteutettu yhdellä sivulla, joka listaa kaikki kyselyt, joihin käyttäjällä on Salesforcessa oikeus. Kyselylistaus näyttää yhdessä listassa sekä mobiililaitteen muistista löytyvät että Salesforcesta haetut kyselyt. Käyttäjä ei voi poistaa tai piilottaa kyselyitä, mutta kyselysivu mahdollistaa kyselyiden haun hakusanan perusteella. Haku hakee kyselyitä mobiililaitteen paikallisesta muistista, jonne kaikki sovelluksessa käsiteltävät kyselyt tallennettaan.

Kyselyn näyttäminen ja kyselyyn vastaaminen

Kuten aiemmin todettiin, kyselyihin on mahdollista vastata anonyymisti, mikä on itse asiassa sovelluksen tämän hetkisessä kehitysversiossa ainoa tapa. Kyselyyn vastaaminen voidaan aloittaa kyselyn tietoja listaavalla sivulla painamalla kyselyn avauspainikkeita. Käynnistyksen yhteydessä voidaan valita käytettäväksi niin kutsuttu messu-tila. Tämän tilan tarkoitus on estää navigointi kyselysivulta pois. Kyselysivulta poistuminen on mahdollista vain käyttäjän asettaman PIN-koodin avulla.

Vastauksen näyttäminen

Kyselyiden vastausten tarkasteleminen mahdollistettiin näyttämällä lista vastauksista kunkin kyselyn tietoja listaavan sivun alla. Lista koostuu kyselyyn vastanneiden henkilön nimistä, joita klikkaamalla päästään katsomaan kyseisen henkilön vastauksia.

Vastausten tallentaminen laitteelle ja vieminen Salesforceen

Oleellinen osa sovelluksen toimintaa oli, että kyselyiden vastaukset saatiin tallennettua laitteen muistiin ja lopulta vietyä Salesforceen.

5 Testaus

Ohjelmistotestauksen tarkoituksena on varmistaa, että ohjelma toimii niin kuin se on suunniteltu toimivan. Testauksen avulla voidaan esimerkiksi havaita koko ohjelmaan tai esimerkiksi johonkin tiettyyn ohjelman osaan, kuten metodiin liittyviä ongelmia. Testauksen avulla voidaan pohjimmiltaan tutkia ohjelman laatua ja tämän perusteella tehdä toimenpiteitä laadun parantamiseksi. Käytännön ohjelmistotestausta ei ole sidottu mihinkään kaavaan, joskin tietyillä testaustavoilla tähdätään esimerkiksi testilähtöiseen ohjelmistokehitykseen.

Fluido Connect Mobilen kehityksessä hyödynnettiin löyhästi niin kutsuttua BDD-ohjelmistokehitysmallia, joka perustuu testilähtöiseen ohjelmistokehitysmalliin (TDD). Testilähtöisessä ohjelmistokehityksessä sovelluskehitys aloitetaan testien kirjoittamisella, joiden perusteella aletaan kirjoittamaan ohjelmistokoodia, joka läpäisee testit. BDD eli käyttäytymislähtöinen ohjelmistokehitys ei vaadi TDD:ssä käytettävien pikkutarkkojen testien kirjoittamista, vaan se tarjoaa laajemman näkökulman ohjelman toiminnan varmistamiseen. Käyttäytymislähtöisessä ohjelmistokehityksessä varmistetaan pikemminkin, miten tietyt ohjelman osat käyttäytyvät. Käyttäytymisenä voidaan pitää esimerkiksi tässä työssä määriteltyjä käyttötapauksia. Käyttötapaukset voidaan edelleen kirjoittaa auki niin sanotuiksi asiakastarinoiksi, jotka voidaan muodostaa BDD:n ehdottaman kaavan mukaan. Koodiesimerkissä 5 on esitetty kyselyjen hakuun hakusanalla liittyvä käyttötapaus BDD:n mukaisena asiakastarinana. Koodiesimerkin riveillä 1-5 esitellään asiakastarina ja riveillä 7-11 määritellään tarina sellaiseen muotoon, jota voidaan hyödyntää suoraan testejä kirjoittaessa. Määrittelyssä käytetään ”given”-, ”when”- ja ”then”-termejä, joilla ilmaistaan vaadittu tilanne testin alussa, testissä tarkasteltava toiminto ja testin odotettu lopputulos. [24; 25.]

```
1.  Tarina (Story): Käyttäjä etsii kyselyitä hakusanan perusteella.
2.
3.  (As a) Käyttäjänä
4.  Haluan (I want to) hakea kyselyitä hakusanalla
5.  Jotta (So that) halutun kyselyn löytäminen olisi helpompaa
```

```
6.
7.  Tapaus 1 (Scenario 1):
8.  (Given) Käyttäjä on kyselyt listaavalla sivulla
9.  Ja (And) käyttäjä on autentikoitunut
10. Kun (When) käyttäjä syöttää hakukenttään tekstiä
11. Sitten (Then) näytä kyselyt, jotka sisältävät hakusanan
```

Koodiesimerkki 5. BDD:n mukainen asiakastarina.

Asiakastarinoiden testaamiseen käytännössä vaaditaan aina tarkoitukseen sopiva ohjelmisto. Tässä työssä käytettiin testien muodostamiseen Jasmine-ohjelmistokehystä ja testin ajamiseen Karma-nimistä ohjelmaa. Jasmine on suunniteltu nimenomaan BDD:n mukaiseen ohjelmistokehitykseen ja sen avulla voidaan testata JavaScript-ohjelmakoodia. Karman avulla puolestaan voidaan ajaa Jasminella kirjoitettuja testejä. Testit voidaan ajaa suoraan selaimessa ja niiden ajaminen voidaan automatisoida, eli testien ajo voidaan käynnistää esimerkiksi aina silloin, kun lähdekoodiin tehdään muutoksia [27]. Koodiesimerkki 6 kuvaa Jasminella kirjoitettua testiä, joka testaa kyselyiden hakua hakusanalla. ”describe”-metodin sisään on kirjoitettu kuvaus siitä, mitä metodi testaa, sekä itse toteutus. Metodin sisältö koostuu useista ”beforeEach”-metodeista, joiden sisällä sovellus alustetaan testin vaatimalla tavalla. Näiden metodien sisällä voidaan luoda toimintaa, joka matkii ohjelman toimintaa tarpeen vaatimalla tavalle. Tällainen tarve on esimerkiksi kyselyiden haku puhelimen muistista tai REST-rajapinnan kautta, mikä ei suoranaisesti liity logiikkaan, jolla kyselyitä haetaan hakusanalla. Testin haluttu lopputulos määritellään ”it”-metodin sisällä. Tällä tavoin kirjoitettujen testien tulokset voidaan esittää suoraan esimerkiksi käyttötapaukset laatineelle henkilölle ja näin varmistaa, että ohjelma toimii halutulla tavalla. [28.]

```
1.  describe("Käyttäjä etsii kyselyitä hakusanan perusteella", function() {
2.     var scope, surveyService, $q;
3.     function initController() {
4.        return controllerFactory('UsersCtrl', {
5.           $scope: scope,
6.           survey: surveyService
7.        });
8.     };
9.     beforeEach(module('fcm'));
10.    beforeEach(inject(function($rootScope, $controller, _$q){
11.       scope = $rootScope.$new();
12.       scope.searchTerm = 'testi';
13.       $controller('surveysCtrl', {$scope: scope});
14.       $q = _$q;
15.    }));
16.    beforeEach(function() {
17.        surveyService.surveys = jasmine.createSpy('surveys');
18.        surveyService.refreshSurveys =
                    jasmine.createSpy('refreshSurveys');
```

```
19.        var defer = $q.defer();
20.          defer.resolve([{'id':1,'name':'First
                    survey'},{'id':2,'name':'Testi kysely'}]);
21.           surveyService.surveys.andReturn(defer.promise);
22.       });
23.    it("Löytyneitä kyselyitä tulisi olla yksi", function(){
24.       initController();
25.       scope.$digest;
26.       expect(scope.surveys.length).toBe(1);
27.    });
28.  });
```

Koodiesimerkki 6. BDD-testi kyselyiden hakemiseen hakusanalla. [26.]

Kuten tämän luvun alussa mainittiin, Fluido Connect Mobilen kehityksessä noudatettiin löyhästi BDD-ohjelmistokehitysmallia. Kehitys ei siis aina tapahtunut testilähtöisesti, vaan testejä kirjoitettiin myös jälkeenpäin. Tämän hetkisessä toteutuksessa ei myöskään testattu Phonegap-sovellusta kokonaisuudessaan, vaan lähinnä HTML5-sovellusta, jonka kehitys tapahtui pääosin tavallisessa työpöytäympäristössä (Windows/OSX). Jatkokehityksen kannalta testiympäristö tulisi luoda myös suoraan mobiilikäyttöjärjestelmillä toimivaksi.

6 Yhteenveto

Tämän insinöörityön tarkoituksena oli toteuttaa alustariippumaton mobiilisovellus Fluido Connect -kyselytyökalun pohjalta. Sovellus toteutettiin HTML5-hybridisovelluksena, koska sen nähtiin olevan tehokkain ja joustavin tapa toteuttaa monella eri alustalla toimiva sovellus. Sovelluskehitystä oli tarkoitus jatkaa aiemman kehitysversion pohjalta, mutta sen puutteellisen toteutuksen myötä sovellus päätettiin kirjoittaa uudelleen eri teknologioita käyttäen. Aiempaan kehitysversioon liittyvää dokumentaatiota voitiin kuitenkin hyödyntää pienin muutoksin tässä työssä.

Sovelluskehityksessä käytetyt teknologiat valittiin sen mukaan, miten niitä oltiin aiemmin Fluidolla käytetty ja millainen tarve uusien teknologioiden tutkimiselle oli. Esimerkiksi HTML5-hybridisovelluksista yrityksellä oli aiempaa kokemusta ja tämän työn yhteydessä niiden toiminta voitiin todistaa toimivaksi myös hieman isommissa, Fluido Connect Mobilen kaltaisessa sovelluksissa. AngularJS, jonka käyttöä ei ole aikaisemmin tutkittu Fluidolla, osoittautui erittäin toimivaksi ratkaisuksi HTML5-sovellusten kehityksessä. AngularJS onkin tässä työssä tehdyn tutkimustyön perusteella voitu nostaa yhdeksi tulevaisuuden projekteissa käytettäväksi

sovelluskehykseksi. Salesforcen kanssa keskustelevan mobiilisovelluksen arkkitehtuurista saatiin myös tärkeää tutkimustietoa ja esimerkiksi BULK-rajapinnan kautta tapahtuva tiedontallennus todettiin toimivaksi. Fluido Connect Mobilen kannalta BULK-rajapinnan käyttö voitiin todeta hieman ylimitoitetuksi, koska käsiteltävä tietomäärä jää käytännön tilanteissa sen verran pieneksi. Pelkkää REST-rajapintaa käyttämällä oltaisiin myös voitu hyödyntää AngularJS:n tarjoamia tekniikoita käsiteltävän tiedon abstraktointiin.

Vaikka tämän työn yhteydessä saatiin tärkeää tietoa HTML5-hybridisovelluksen kehittämisestä, jäi sovelluksen toteutus hieman vaillinaiseksi. Keskeisimmät toiminnallisuudet saatiin toteutettua, mutta täydellinen ja käyttäjäkokemukseltaan kypsä sovellus jäi vielä jatkokehityksen vastuulle. Jatkokehityksen yhteydessä on tarkoitus toteuttaa puuttuva toiminnallisuus (esimerkiksi kyselyjen vastausten vertailun) ja toteuttaa käsiteltävän tiedon korkeampi abstraktiotaso, joka helpottaa muun muassa testien kirjoittamista. Testejä varten testiympäristö on sovitettava yhteen mobiililaitteiden käyttöjärjestelmien kanssa, jotta sovelluksen laadusta ja valmiusasteesta voidaan jatkossa tehdä tarkempia päätelmiä.

Lähteet

1 Fluido Oy. Verkkodokumentti. http://www.fluido.fi/fluido.html, Luettu 15.3.2014.

2 Fluido Oy. Verkkodokumentti. http://www.fluido.fi/salesforce-laajennukset.html. Luettu 15.3.2014.

3 Lehdonvirta Pyry & Korpela Jukka K. 2013. HTML5 sovellusalustana.

4 Whitley Richard & Korf Mario. 2013. Mobile App Developer Guide.

5 Pilgrim Mark. 2010. HTML5: Up & Running. Sebastopol, California: O'Reilly Media, Inc.

6 Ovatko HTML5-hybridisovellukset liian hitaita? Blogi. http://geniem.fi/onko-html5-hybridi-sovellukset-liian-hitaita/. Luettu 28.3.2014.

7 Native, HTML5, or Hybrid: Understanding Your Mobile Application Development Options. Verkkodokumentti. https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options. Luettu 20.4.2014.

8 Salesforce.com. Verkkodokumentti. http://en.wikipedia.org/wiki/Salesforce.com. Luettu 6.4.2014.

9 Salesforce Product Overview. Verkkodokumentti. http://help.salesforce.com/apex/HTViewHelpDoc?id=overview.htm. Luettu 6.4.2014.

10 Understanding Force.com REST Resources. Verkkodokumentti. https://www.salesforce.com/us/developer/docs/api_rest/Content/intro_rest_resources.htm. Luettu 6.4.2014.

11 Overview, Phonegap. Verkkodokumentti. http://docs.phonegap.com/en/edge/guide_overview_index.md.html#Overview. Luettu 15.4.2014.

12 Streamline Cross-Platform Development Using Apache Cordova / PhoneGap CLI. Verkkodokumentti. http://blog.safaribooksonline.com/2013/07/19/streamline-cross-platform-development-using-apache-cordova-phonegap-cli/. Luettu 15.4.2014.

13 AngularJS introduction. Verkkodokumentti. http://docs.angularjs.org/guide/introduction. Luettu 28.3.2014.

14 Bootstrap. Verkkodokumentti. http://getbootstrap.com/getting-started/. Luettu 1.4.2014.

15 CSS-tricks. How To Create an IE-Only Stylesheet. Blogi. http://css-tricks.com/how-to-create-an-ie-only-stylesheet/. Luettu 1.4.2014.

16 Dependency Injection, Verkkodokumentti. http://docs.angularjs.org/guide/di. Luettu 5.4.2014

17 Introduction. Bulk API. Verkkodokumentti. https://www.salesforce.com/us/developer/docs/api_asynch/Content/asynch_api_intro.htm. Luettu 5.4.2014

18 Introducing Force.com REST API. Verkkodokumentti. https://www.salesforce.com/us/developer/docs/api_rest/Content/intro_what_is_rest_api.htm. Luettu 5.4.2014

19 Conceptual Overview, AngularJS. Verkkodokumentti. https://docs.angularjs.org/guide/concepts. Luettu 21.4.2014.

20 Autentikoinnin perusteet. Verkkodokumentti. https://sec.cs.tut.fi/maso/materiaali.php?id=61. Luettu 10.4.2014

21 Digging Deeper into OAuth 2.0 on Force.com. Verkkodokumentti. https://developer.salesforce.com/page/Digging_Deeper_into_OAuth_2.0_on_Force.com. Luettu 11.4.2014.

22 Promise & Deferred objects in JavaScript Pt.1: Theory and Semantics. Chris Webb. Blogi. http://blog.mediumequalsmessage.com/promise-deferred-objects-in-javascript-pt1-theory-and-semantics. Luettu 21.4.2014.

23 Same-origin policy. Verkkodokumentti. https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy. Luettu 15.4.2014.

24 Behavior-driven development. Verkkodokumentti. http://en.wikipedia.org/wiki/Behavior-driven_development. Luettu 29.4.2014.

25 INTRODUCING BDD, Dann North. Verkkodokumentti. http://dannorth.net/introducing-bdd/. Luettu 29.4.2014.

26 AngularJs Good Unit Test Structure For Controllers & How to test ajax code and Promises. Nadeem Khedr. Blogi. http://nadeemkhedr.wordpress.com/2013/10/18/angularjs-good-unit-test-structure-for-controllers/. Luettu 29.4.2014.

27 Karma - Test Runner for JavaScript. Vojta Jina. Verkkovideo. https://www.youtube.com/watch?v=YG5DEzaQBIc. Katsottu 29.4.2014.

28 Jasmine. Verkkodokumentti. http://jasmine.github.io/2.0/introduction.html. Luettu 30.4.2014.